Uma alternativa colaborativa para expansão de dicionários morfológicos de tradutores automáticos baseados em regras

Vinícius Silva Nogueira[1] e Aléssio Miranda Júnior[2]
Departamento de Computação e Construção Civil
Centro Federal de Educação Tecnológica de Minas Gerais
Timóteo, Brasil
[1]*vinicius_snogueira@hotmail.com*
[2]*alessio@cefetmg.br*

***Resumo*—Neste trabalho, apresentamos a proposta de um ambiente *web* colaborativo, chamado Apertium WDM, para ampliação de dicionários morfológicos utilizados em tradutores automáticos baseados em regras, mais especificamente para o tradutor Apertium. O ambiente disponibiliza uma interface homem-máquina que permite com que usuários não especialistas em computação e com conhecimentos básicos sobre uma língua possam contribuir com a expansão do vocabulário de um dicionário morfológico. Tal interface faz uso de um conjunto de funcionalidades para manipulação destes dicionários, que são fornecidas por uma API, e todas as contribuições são armazenadas em repositórios Git. Com esse ambiente, espera-se reduzir o conhecimento necessário como pré-requisito para uma pessoa se tornar apta a contribuir com a expansão de um dicionário morfológico e, com isso, aumentar o número de contribuidores em potencial do Apertium.**

***Keywords*-tradução automática; Apertium; dicionário morfológico; desenvolvimento colaborativo;**

I. Introdução

O processamento de linguagens naturais (PLN) é uma subárea da ciência da computação que tem por objetivo aprender, entender e produzir conteúdos de linguagens naturais. O PLN vem crescendo nos últimos 20 anos tanto como uma área de pesquisa científica quanto uma área de aplicações práticas [1]. Um dos principais desafios da computação para as próximas décadas é desenvolver sistemas capazes de processar com eficiência as linguagens naturais [2].

Dentro do campo de PLN, destaca-se a tradução automática (TA), que é uma subárea da linguística computacional que tem como objetivo a tradução automática de textos de uma língua natural de origem para uma língua natural de destino, traduzindo todo seu conteúdo de maneira a gerar um resultado inteligível e que preserve as características do texto original, como estilo, coesão e significado [3].

Os sistemas de TA são cada vez mais necessários devido à natureza multilíngue da sociedade globalizada e das complexas redes sociais, organizacionais e comerciais existentes [2]. A comunicação entre povos distintos facilitada por meios eletrônicos é uma realidade e com evoluções cadas vez mais esperadas. Contudo, os sistemas de tradução mais bem sucedidos têm sido tradicionalmente *softwares* proprietários, os quais utilizam tanto componentes internos quanto bases de dados fechadas.

Alguns sistemas de TA considerados gratuitos estão disponíveis para uso na internet com algumas restrições. Essas ferramentas são distribuídas com propósitos comerciais e não podem ser adaptadas ou aprimoradas para uma finalidade específica [4]. Essa forma de distribuição dificulta a inserção de suporte a novos idiomas, que na maioria das vezes é feita objetivando algum retorno financeiro, e de sua utilização em outras aplicações.

Os sistemas de PLN em geral estão disponíveis apenas para línguas com alta quantidade de *corpus* linguísticos disponíveis, como o Inglês, Francês, Espanhol, Alemão e Chinês. Em contrapartida, diversas línguas com baixa quantidade de *corpus* linguísticos, como Bengali, Indonésio, Punjabi, Cebuano e Swahili, que são faladas e escritas por milhões de pessoas, não possuem tais sistemas disponíveis [1].

Dessa forma, surgiram diversos projetos *free/open-source* que buscam na forma livre e colaborativa de desenvolvimento superar tais dificuldades. Dentre esses projetos, destaca-se o Apertium[1], que é uma plataforma, licenciada sob a *GNU General Public License*[2], para desenvolvimento de máquinas de tradução automática baseadas em regras (*Rule-Based Machine Translation* ou RBMT) e que será o foco do presente trabalho.

As RBMT dependem de dados linguísticos explícitos, como dicionários morfológicos, dicionários bilíngues, gramáticas e regras de transferência estrutural [5]. A base de conhecimento linguístico do Apertium é formada por aquivos XML com formatos bem definidos para diversos pares de línguas.

O Apertium oferece suporte a mais de quarenta línguas e o desenvolvimento de novos pares de tradução vem sendo alvo de diversos trabalhos nos últimos anos, como podemos ver em Tyers et al. (2017) o par italiano-sardenho, em Johnson et al. (2017) o par sámi-finlandês e em Klubička, Ramírez-

[1]https://www.apertium.org
[2]http://www.gnu.org/licenses

Sánchez e Ljubešić (2016) o par croata-sérvio. Suas bases de conhecimentos são mantidas em repositórios Git públicos. Apesar de serem colaborativas, existem dificuldades para expandi-las, uma vez que o tamanho e a complexidade dos dicionários, juntamente com as particularidades de cada idioma, tornam essa tarefa árdua e demorada.

Na maior parte das línguas latinas e germânicas, por exemplo, as palavras compartilham um pequeno grupo de padrões de flexão. Estes padrões são denominados paradigmas e a maior parte deles são definidos nos estudos iniciais para a criação do dicionário morfológico. A partir de então, a expansão destes dicionários consiste basicamente na inserção de novas palavras e na sua respectiva associação com um paradigma, sendo raramente necessário a criação de um novo paradigma específico para uma palavra. Esta tarefa pode ser executada por usuários leigos em computação que possuem um conhecimento mínimo sobre a língua, desde que exista uma interface adequada para isso.

Dentre as principais limitações das RBMT, em geral, estão o alto custo no desenvolvimento dos recursos linguísticos e na sua extensão para outros idiomas [9]. De fato, a ausência de uma interface específica para manutenção de seus dicionários morfológicos, de pares e regras de tradução é um dos principais obstáculos para o crescimento do Apertium. As poucas alternativas de interface ainda visam usuários especialistas tanto em computação quanto em linguística [3].

Atualmente, o Apertium conta com poucas dezenas de especialistas, sendo que o número de usuários (que são colaboradores em potencial) é da ordem de milhares. A necessidade de conhecimentos em linguagens de marcação (XML), sistemas de controle de versões (Git) e das estruturas internas do Apertium, acabam se tornando uma barreira que limita o número de colaboradores aptos a contribuir e afasta colaboradores em potencial.

Dessa forma, este trabalho tem como objetivo principal propor um ambiente *web* colaborativo - o Apertium Web Dictionary Maintenance ou Apertium WDM - para o aumento do vocabulário dos dicionários monolíngues do Apertium, a fim de reduzir a quantidade de conhecimento necessário como pré-requisito para uma pessoa se tornar apta a contribuir, o que poderá aumentar o número de contribuidores e, por conseguinte, a capacidade de tradução do Apertium.

O Apertium WDM proverá uma interface homem-máquina que permite que usuários sem conhecimentos específicos em computação e com conhecimentos básicos sobre uma língua possam contribuir com um dicionário. Todas alterações são enviadas a um repositório Git e são julgadas pelos usuários especialistas mantenedores das bases de conhecimento.

Também, objetivam-se mais especificamente:

1) Especificar uma API RESTful que forneça um conjunto de funcionalidades para adição, de forma colaborativa, de novas palavras aos dicionários morfológicos de línguas de origem latina e germânica do Apertium, que são armazenados em um repositório Git;
2) Especificar uma GUI (*Graphical User Interface*) que permita ao usuário utilizar de forma prática os recursos fornecidos pela API;
3) Propor um fluxo de trabalho colaborativo entre os contribuidores e os mantenedores das bases de dados.

O restante deste trabalho está organizado da seguinte forma. A Seção II apresenta os conceitos fundamentais da tradução automática. A Seção III apresenta os módulos internos e a estrutura dos dicionários do Apertium. A Seção IV apresenta algumas das principais ferramentas utilizadas atualmente para o desenvolvimento colaborativo das bases de conhecimento do Apertium. A Seção V apresenta o ambiente proposto e, por fim, a Seção VI apresenta as considerações finais deste trabalho.

II. Linguagem natural e tradução automática

A linguagem pode ser definida como uma ferramenta de comunicação que permite a troca de informações entre dois indivíduos. A linguagem natural (LN) é a forma mais comum de comunicação e origina-se na capacidade inata de comunicação do ser humano. Ela é representada, por exemplo, pelas linguagens faladas, como o português, e pelas linguagens de sinais, como a Linguagem Brasileira de Sinais (Libras).

Atualmente, evidencia-se cada vez mais a necessidade da tradução entre LN devido à globalização e à natureza cada vez mais multicultural da sociedade. Mais de 7000 línguas já foram catalogadas ao redor do mundo. No Brasil, 237 já foram catalogadas, sendo que, destas, 217 estão vivas e 20 já são consideradas extintas [10].

A automatização do processo de tradução (tradução automática) vem sendo um dos principais desafios da computação nas últimas décadas. Os maiores empecilhos para uma tradução automática eficiente se devem à natureza dinâmica e ambígua das LN, que são consideras objetos vivos e que estão em constante mudança.

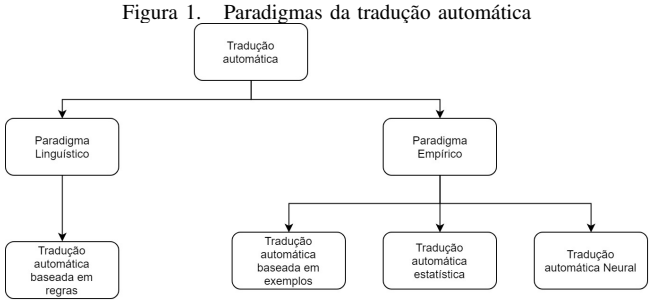
Desde o início dos estudos da TA até os dias atuais, as principais estratégias utilizadas no processo de tradução para gerar um texto na língua alvo são a tradução direta, a tradução por transferência e a tradução por interlíngua [3] [9].

Na tradução direta, as unidades lexicais do texto na língua de origem são diretamente mapeadas para unidades lexicais correspondentes na língua de destino sem que haja nenhum processamento intermediário, sendo considerado o processo mais simples e que produz resultados pouco satisfatórios.

A tradução por transferência, por sua vez, realiza uma análise sintática parcial ou completa da língua fonte e o mapeamento fonte-alvo se dá com base em regras de transferência sintática [9].

Por fim, temos a tradução por interlíngua, que realiza uma representação completa do texto na língua de origem para

Figura 1. Paradigmas da tradução automática



uma língua intermediária e, então, é gerado o texto na língua alvo a partir do texto na língua intermediária.

Os sistemas de TA são, ainda, classificados quanto ao paradigma utilizado (vide Figura 1), sendo eles: o paradigma linguístico, onde o conhecimento das línguas de origem e de destino são mapeados na forma de regras, como ocorre nas RBMT, e o paradigma empírico, onde a tradução é realizado sobre a análise de *corpus* bilíngues e que engloba a tradução automática estatística (*Statistical Machine Translation* ou SMT), a tradução automática baseada em exemplos (*Example-based Machine Translation* ou EBMT) e a recente tradução automática neural (*Neural Machine Translation* ou NMT) [9].

Estes quatro modelos teóricos de tradução são discutidos a seguir.

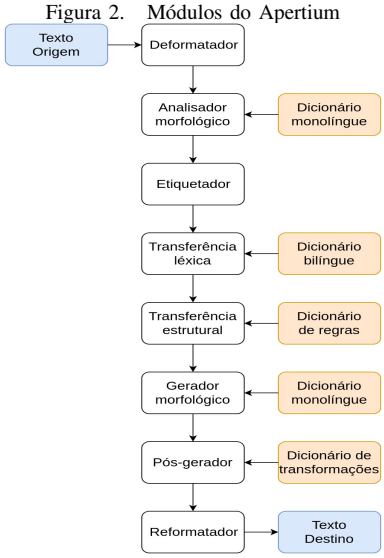
- **Tradução automática estatística:** Na SMT as traduções são geradas a partir de modelos estatísticos cujos parâmetros são derivados da análise de *corpus* linguísticos bilíngues. Essa paradigma era considerado o estado da arte e o Google o utilizava em seu tradutor (Google Translator) até 2016, quando deu lugar a NMT [9]. SMT procuram balancear a probabilidade de que a palavra da sentença traduzida corresponda à palavra da sentença original (fidelidade) e que as palavras da sentença traduzida são essas e que ocorrem nessa ordem no idioma alvo (fluência) [11]. Dentre suas principais vantagens estão a produção de uma tradução mais fluente, na maioria das vezes, quando comparado a RBMT [11] e a independência quanto ao idioma utilizado, uma vez que é possível gerar tradutores para diferentes idiomas utilizando os mesmos algoritmos. Em contrapartida, a SMT não é aplicável caso não existam *corpus* suficientes para a língua em questão, a qualidade da tradução pode ser imprevisível e existe uma dificuldade em interpretar e modificar os valores gerados pelo modelo estatístico [9].
- **Tradução automática baseada em exemplos:** Assim como a SMT, a EBMT utiliza a análise de *corpus* linguísticos bilíngues para realizar suas traduções. Porém, ao contrário da SMT, que de forma resumida busca combinar palavras ou frases com maior probabilidade, a EBMT busca encontrar padrões de tradução nos fragmentos apresentados como exemplos.
- **Tradução automática baseada em regras:** Por sua vez, as RBMT abordam o paradigma baseado em informações linguísticas e contam com inúmeras regras de tradução incorporadas a diversos dicionários disponíveis para cada par de tradução. As traduções são geradas a partir de regras morfológicas, sintáticas e semânticas além de regras de transferência de um idioma para outro. A criação manual das regras é, ao mesmo tempo, uma das principais vantagens e desvantagens dessa abordagem. A possibilidade de definição explícita de cada regra permite ao projetista do tradutor tratar de forma eficiente casos específicos de tradução, mas em contrapartida, a criação de todas as regras de tradução demandam um grande esforço humano.
- **Tradução automática neural:** A tradução utilizando redes neurais artificiais (RNA), por fim, é uma nova abordagem que surgiu nos últimos anos que objetiva o treinamento de uma RNA para a tradução entre dois idiomas. Ao contrário dos outros métodos de tradução, que são subdivididos em diversos componentes, esse método visa a construção e treinamento de uma única e grande RNA que lê uma sentença e gera sua tradução [12]. As principais vantagens em relação aos modelos tradicionais são os poucos conhecimentos linguísticos necessários para gerar o tradutor, a otimização conjunta de toda a rede e não apenas de módulos específicos e o fato do conhecimento gerado ser mais compacto, do que, por exemplo, os dicionários das RBMT [9]. Contudo, esse modelo costuma ser menos preciso que os modelos estatísticos (até então considerado estado da arte). Isso ocorre por três motivos principais: o processo de treinamento e de tradução são lentos; é ineficaz em lidar com palavras raras; e algumas vezes falha ao traduzir todas as palavras de uma sentença [13]. Esses problemas têm impedindo que a tradução utilizando RNA seja utilizada em serviços e implementações práticas, onde a velocidade e precisão são essenciais [13].

III. Apertium

O Apertium é uma plataforma de tradução livre (FOSS) resultante do projeto "Máquina de Tradução Automática Open-Source para as línguas da Espanha" (*"Traducción automática de código abierto para las lenguas del estado español"*) desenvolvido na Universidade de Alicante e financiado pelo governo espanhol e catalão, a fim de criar uma MTA entre as línguas da Espanha. Ele, o Apertium, é um *shallow-transfer rule-based machine translation*[3] desenvolvido, inicialmente, para traduzir idiomas semelhantes. Com o passar dos anos, notou-se a eficiência de sua tradução e a partir de então sua base de conhecimento vem sendo expandida para dezenas de outros pares de línguas distantes.

[3]Máquina de tradução automática baseada em regras de transferências superficiais

Figura 2. Módulos do Apertium



O mecanismo de tradução do Apertium é composto por 8 módulos e utiliza quatro dicionários de dados que estão representados na Figura 2 e que estão descritos nas subseções seguintes.

A. *Especificações das bases de dados: Dicionários*

1) Dicionário monolíngue: Os dicionários monolíngues (ou dicionários morfológicos ou ainda DM) são utilizados no processo de análise morfológica e também no processo de geração morfológica. Eles são compostos pela definição de um alfabeto e de um conjunto de símbolos, paradigmas e lemas.

- **Definição do alfabeto:** Consiste em definir o conjunto de caracteres que são utilizados pela língua. O alfabeto é utilizado na etapa de análise morfológica e permite ao analisador gerar corretamente os *tokens* das palavras do texto de entrada. Sua definição é delimitada pelo elemento <*alphabet*>, como no Exemplo 1 que mostra a definição do alfabeto utilizado no DM de português.

```
<alphabet>
    ÀÁÂÃÄÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜàáâãäçèéê
    ëìíîïñòóôõöùúûüABCDEFGHIJKLMNOPQR
    STUVWXYZabcdefghijklmnopqrstuvwxy
    z
</alphabet>
```

Exemplo 1. Alfabeto do dicionário morfológico de português

- **Definição dos símbolos:** Consiste na definição das unidades gramaticais utilizados pelo dicionário. A definição dos símbolos é delimitada pelo elemento <sdefs> e cada símbolo é representado pelo elemento <sdef>. O atributo "n" define o nome do símbolo. O Exemplo 2 mostra a definição de alguns símbolos utilizados no DM de português sendo eles: substantivos (n, do inglês *noun*), nomes pessoais (np), adjetivos (adj), feminino (f), masculino (m), singular (sg), plural (pl) e adverbio (adv).

```
<sdefs>
    <sdef n="n"/>
    <sdef n="np"/>
    <sdef n="adj"/>
    <sdef n="f"/>
    <sdef n="m"/>
    <sdef n="sg"/>
    <sdef n="pl"/>
    <sdef n="adv"/>
          ...
</sdefs>
```

Exemplo 2. Símbolos do dicionário morfológico de português

- **Definição dos paradigmas:** Em grande parte das línguas, muitos lemas compartilham o mesmo padrão de flexão. Utiliza-se então paradigmas, que são padrões de flexões comuns à várias palavras, para agrupá-las e evitar-se, então, escrever todas as flexões de todas as palavras. A definição dos paradigmas é delimitada pelo elemento <pardefs> e cada paradigma é representado por um elemento <pardef>.
O Exemplo 3 mostra a definição do paradigma "abadia" do dicionário de português, que é utilizado por 6968 lemas.

```
<pardefs>
          ...
     <pardef n="abadia__n">
          <e><p><l>s</l><r><s n="n"/>
          <s n="f"/><s n="pl"/></r></
          p></e>
          <e><p><l/><r><s n="n"/><s n
          ="f"/><s n="sg"/></r></p></
          e>
    </pardef>
          ...
</pardefs>
```

Exemplo 3. Paradigmas do dicionário morfológico de português

A Tabela I descreve os demais elementos utilizados na definição de um paradigma. É adotado a abordagem de pares de direção, pois posteriormente o dicionário será compilado para uma máquina de estados finitos e será utilizado tanto no processo de análise morfológica (direção esquerda para direita) quanto no processo de geração morfológica (direçã direita para esquerda), como descrito nas Seções III-B2 e III-B6.

Elemento	Descrição
e	Define o início de uma nova entrada
p	Define o início de novo par de direções
l	Do inglês *left*, define a entrada esquerda
r	Do inglês *right*, define a entrada direita
s	Define o símbolo associado a entrada direita

Tabela I
ELEMENTOS DA DEFINIÇÃO DE PARADIGMAS

A Tabela II mostra o processo de análise e geração morfológica utilizando o paradigma definido no Exemplo 3.

Entrada	Direção	Saída
abadia	esquerda para direita	abadia<n><f><sg>
abadias	esquerda para direita	abadia<n><f><pl>
abadia<n><f><sg>	direita para esquerda	abadia
abadia<n><f><pl>	direita para esquerda	abadias

Tabela II
PROCESSO DE ANÁLISE E GERAÇÃO MORFOLÓGICA

- **Definição dos lemas:** Nessa seção ocorre a definição de todos os lemas presentes na língua e suas respectivas associações com o paradigma utilizado. Eles são agrupados em seções que são delimitadas pelo elemento <section>. O Exemplo 4 mostra a definição de alguns lemas do DM de português. Cada lema é representado pelo elemento <e>. O atributo "lm" define o nome que representa esse lema e os elementos <i> e <par> definem, respectivamente, a parte invariante do lema e o paradigma associado a ele.

```
<section id="main" type="standard">
        ...
    <e lm="contingência"><i>conting
    ência</i><par n="abadia__n"/></e>
    <e lm="advertência"><i>advert
    ência</i><par n="abadia__n"/></e>
    <e lm="adaptabilidade"><i>
    adaptabilidade</i><par n="
    abadia__n"/></e>
    <e lm="fraternizar"><i>fraterniz<
    /i><par n="abal/ar__vblex"/></e>
    <e lm="reciclar"><i>recicl</i><
    par n="abal/ar__vblex"/></e>
        ...
</section>
```

Exemplo 4. Lemas do dicionário morfológico de português

2) Dicionário bilíngue: Os dicionários bilíngues (DB) armazenam os mapeamentos de tradução de cada lema de uma língua para outra. Eles são compostos pela definição dos símbolos utilizados nas duas línguas e por uma única seção que realiza a associação dos lemas.

O Exemplo 5 mostra como seria a estrutura de um dicionário bilíngues português-espanhol. Dentro da seção principal, é dado o exemplo de associação do lema em português "menino" ao seu correspondente "*niño*" em espanhol.

```
<dictionary>
    <sdefs>
        ...
        <sdef n="n"/>
        <sdef n="np"/>
        <sdef n="adj"/>
        <sdef n="f"/>
        <sdef n="m"/>
        <sdef n="mf"/>
        <sdef n="sg"/>
        <sdef n="pl"/>
        <sdef n="adv"/>
        ...
    </sdefs>
    <section id="main" type="standard">
        ...
        <e><p><l>menino<s n="n"/></l><r>
        niño<s n="n"/></r></p></e>
        ...
    </section>
</dictionary>
```

Exemplo 5. Exemplo dicionário bilíngue português-espanhol

3) Dicionário de regras: Os dicionários de regras armazenam as regras de transferência necessárias para a tradução, sendo necessário um dicionário para cada sentido de tradução. As regras são formadas por padrões de sequência em que categorias lexicais aparecem no texto de entrada [14]. Ao serem detectados, os ajustes definidos por essa regra serão aplicados no texto de saída. Esse dicionário não será abordado neste trabalho.

4) Dicionário de transformações: Os dicionários de transformações armazenam as regras para modificação das palavras e é utilizado após a etapa de geração morfológica. Tais modificações são, por exemplo, contrações das palavras, acréscimo de apóstrofe e outros modificações que uma palavra pode sofrer caso haja uma outra palavra específica próxima a ela. Assim como o dicionário de regras, esse dicionário também não será abordado neste trabalho.

B. Arquitetura do Apertium

1) Deformatador: O deformatador é o módulo responsável por separar o texto a ser traduzido das informações de formato (HTML, RTF, etc.) em que ele se encontra [14]. O Exemplo 6 mostra um texto em HTML a ser deformatado.

```
    <header> Texto a ser traduzido </
    header>
```

Exemplo 6. Texto de entrada do deformatador

O Exemplo 7 mostra o texto deformatado.

```
    [<header>] Texto a ser traduzido [</
    header>]
```

Exemplo 7. Texto de saída do deformatador

Todas as informações de marcação do formato são colocadas entre colchetes que serão ignorados pelos módulos subsequentes.

2) Analisador morfológico: O analisador morfológico, como o próprio nome já diz, é responsável por realizar a análise morfológica das palavras na forma superficial (FS), ou seja, a forma em que a palavra aparece no texto e gerar seus correspondentes na forma léxica (FL). A forma léxica, por sua vez, é composta pelo lema da palavra e por etiquetas que representam sua categoria (substantivo, adjetivo, pronome, etc.) e suas flexões (gênero, número, tempo verbal, etc.). Cada palavra pode possuir mais de uma forma léxica devido à ambiguidade presente nas LN.

3) Etiquetador: O módulo de etiquetagem é responsável por escolher a FL correta dentre as possíveis alternativas geradas pelo módulo anterior (devido a ambiguidade supracitada). Para tal, ele utiliza uma modelo estatístico (mais especificamente, o modelo oculto de Markov) [14].

4) Transferência léxica: No etapa de transferência léxica é onde ocorre, de fato, a tradução. Nesse módulo é utilizado o dicionário bilíngue e realizado o mapeamento de cada FL do idioma de origem para o idioma de destino. Os lemas de cada FL são trocados pelos lemas correspondentes no idioma alvo e as etiquetas são mantidas (exceto quando exista explicitamente uma indicação para mudá-las).

5) Transferência estrutural: O módulo de transferência estrutural é responsável por processar pequenos trechos ou frases que precisam de um processamento extra devido à divergências gramaticais entre os dois idiomas (mudanças de gênero e número, reordenação de palavras, mudanças de preposições, etc.) [14]. Para tanto, é utilizado o dicionário de regras que diz qual ação deve ser tomada para cada padrão encontrado.

6) Gerador morfológico: O gerador morfológico, por sua vez, utiliza o dicionário monolíngue da língua alvo para gerar as palavras do texto na forma superficial a partir da forma léxica.

7) Pós-gerador: O módulo pós-gerador utiliza um dicionário de transformações para, por exemplo, criar abreviações e contrações no texto traduzido.

8) Reformatador: O reformatador é responsável por converter o texto traduzido para o formatado de entrada, mantendo sua formatação original.

IV. Principais ferramentas

Esta seção apresenta algumas das principais ferramentas utilizadas atualmente para o desenvolvimento colaborativo das bases de dados do Apertium.

A. Git

Git é uma ferramenta distribuída de controle de versões, originalmente projetada para desenvolvimento de *software*, mas que também pode ser utilizada para registrar o histórico de edição de qualquer arquivo de texto. O funcionamento do Git consiste em basicamente na realização e armazenamento de *snapshots*[4] do projeto. Sempre que o usuário realiza um *commit*[5], uma cópia de todos os arquivos do projeto é salva e associada a um *snapshot*. Caso um arquivo não tenha sido alterado, por questões de eficiência, é salvo apenas uma referência para o último *snapshot* em que esse arquivo foi alterado [15].

A utilização de *branches* (ou ramificações) é a forma de desenvolvimento não linear adotada pelo Git. Em cada *commit* é salvo um ponteiro (ou referência) para o *commit* anterior, exceto para o primeiro *commit* realizado e para *commit* originalizados da junção de duas ou mais *branches*, que armazenarão, então, um ponteiro para cada uma delas.

Todo repositório Git é inicializado com uma *branch* principal que é denominada *master*. Uma *branch* é um ponteiro para um *commit* que se move automaticamente para o *commit* recém adicionado. Iniciar uma nova *branch* consiste, então, em criar um novo ponteiro que irá se mover de forma independente dos demais.

O ambiente *web* proposto é conectado a um repositório Git, que será usado como ferramenta administrativa e de controle de versão dos dicionários. Cada *branch* no repositório corresponderá a uma versão do dicionário que está sendo modificada por um usuário, que ao terminar sua contribuição, poderá solicitar sua junção com a *branch* principal. Essa junção, por sua vez, será analisada por um mantenedor do dicionário que julgará se ela deve ser aceita ou não.

B. Dixtools

O Apertium-dixtools[6] é uma ferramenta de código fonte livre, disponível em linha de comando, que dispõe algumas funcionalidades para manipulação dos dicionários do Apertium. Dentre elas, destacam-se:

- **Formatar dicionário:** reestrutura as definições de palavras e suas traduções, formatando, por exemplo, uma palavra por linha.
- **Ordernar um dicionário:** ordena as palavras em ordem alfabética.
- **Leitor de dicionário:** obtém listas dos elementos (lemas, paradigmas, definições...) de um dicionário.
- **Inverter dicionário:** cria um dicionário bilíngue de uma língua A para uma língua B a partir de um dicionário bilíngue da língua B para língua A.
- ***Crossdics*:** constrói um dicionário bilíngue de uma linguagem A para outra linguagem C a partir de um dicionário da linguagem A para linguagem B e de um dicionário da linguagem B para a linguagem C.
- **Cobertura do dicionário:** gera dados estatístico da frequência de uso de diferentes palavras do dicionário.
- ***Autoconcord*:** faz dicionários bilíngues entrarem em concordância com os dicionários monolíngues quando

[4] *Snapshot*: Retrato dos arquivos do sistema em um dado momento.

[5] *Commit*: Salvar o estado atual do projeto. Muitas vezes utilizado como sinônimo de *snapshot*.

[6] http://wiki.apertium.org/wiki/Apertium-dixtools

existem diferenças (gênero, número, etc.) entre as duas linguagens.

- **Paradigmas equivalentes:** utilizado para encontrar paradigmas não utilizados e paradigmas funcionando da mesma forma que outro.
- **Unir dicionários:** permite unir lista de palavras de vários dicionários monolíngues.
- **Aumentar:** permite adicionar novas palavras ao dicionário interativamente.
- ***Grep*:** permite filtrar um dicionário baseado em um lema ou paradigma específico e gerar um novo dicionário.

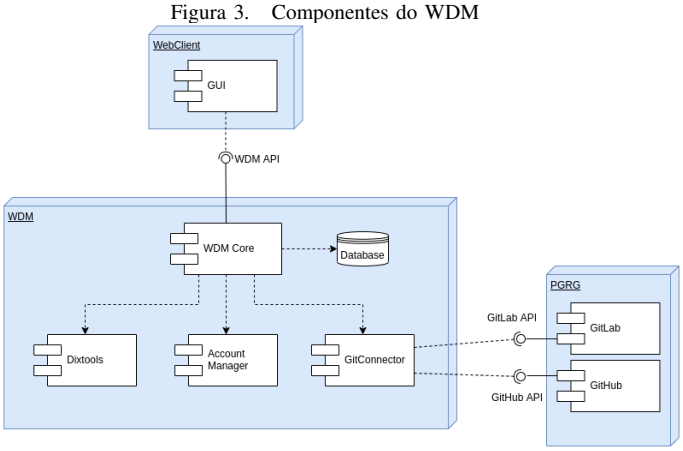
Apesar de fornecer ferramentas úteis, a instalação e utilização do Dixtools é feitas apenas através de linha de comando, o que acaba dificultando sua utilização por pessoas leigas em computação. Além disso, ele permite apenas que os dicionários sejam modificados localmente, sem nenhum tipo de colaboração, sendo para isso necessário a utilização de outras ferramentas externas.

Parte do código fonte do Dixtools que realiza a representação e relacionamento dos elementos de um dicionário, bem como sua leitura e escrita, será reutilizado neste trabalho.

V. Arquitetura

O ambiente proposto tem como principal filosofia proporcionar um meio pelo qual usuários não especialistas em computação possam contribuir com o aumento do vocabulário de um dicionário morfológico, de forma que não seja preciso alterar a estrutura dos dicionários e que possa ser integrado à atual forma de desenvolvimento colaborativo (contribuições de usuários especialistas em repositórios Git). A Figura 3 apresenta uma visão geral do ambiente proposto.

Figura 3. Componentes do WDM



O ambiente pode ser dividido em três módulos: o PGRG (Plataformas Gerenciadoras de Repositórios Git), que é um módulo externo ao projeto e é constituído por plataformas que realizam o gerenciamento de repositórios Git, como o GitLab e o GitHub; o *Web Client*, que disponibiliza uma GUI que permite com que os usuários realizem modificações pontuais sobre os arquivos de texto dos dicionários de forma facilitada, organizada e sem efeitos colaterais; e por fim, o WDM, que disponibiliza uma API com um conjunto de funcionalidades específicas para manipulação dos dicionários (descrita na Seção V-A).

A. Especificação dos recursos da API

Os recursos da API seguem ao padrão de uma API RESTful. Foram utilizadas as operações POST, GET, PUT e DELETE. A seguir, são apresentados os cinco recursos principais.

1) OfficialDictionary: O recurso OfficialDictionary é usado para listagem e sincronização dos dicionários oficiais dos repositórios Git e suas respectivas versões. Cada dicionário oficial possui um repositório distinto e é representado por sua *branch* principal. Diferentes versões do dicionário podem ser utilizadas, sendo que para isso as versões devem estar identificadas por *tags* na *branch* principal. A Tabela III mostra a URI relativa desse recurso e as operações que ele suporta.

URI	POST	GET	PUT	DEL
/officialdictionary		X		
/officialdictionary/{id}		X		
/officialdictionary/synchronize	X			

Tabela III
Operações suportadas pelo recurso OfficialDictionary

No Exemplo 8 é possível visualizar o modelo de resposta da operação GET da URI "/officialdictionary".

```
[
  {
    "id": 0,
    "language": "string",
    "reference": "string",
    "version": "string"
  }
]
```

Exemplo 8. Resposta da operação GET do recurso OfficialDictionary

2) MyDictionary: O recurso MyDictionary permite ao usuário criar sua cópia de um dicionário oficial. Para isso, ele deve indicar o dicionário oficial e a versão desejada e, então, uma nova *branch* exclusiva será criada no repositório do dicionário oficial e armazenará as modificações feitas pelo usuário. O usuário poderá, então, criar uma solicitação de *merge* com a *branch* principal ou fazer *download* do dicionário. A Tabela IV mostra a URI relativa desse recurso e as operações que ele suporta.

No Exemplo 9 é possível visualizar o modelo de resposta da operação GET da URI "/mydictionary".

URI	POST	GET	PUT	DEL
/mydictionary	X	X		
/mydictionary/{id}		X	X	X
/mydictionary/{id}/merge	X			
/mydictionary/{id}/download		X		

Tabela IV
OPERAÇÕES SUPORTADAS PELO RECURSO MYDICTIONARY

```
[
  {
    "id": 0,
    "name": "string",
    "officialDictionaryId": 0,
    "officialDictionaryLanguage": "
    string",
    "qtLemma": 0,
    "qtParadigm": 0,
    "qtSymbol": 0
  }
]
```

Exemplo 9. Resposta da operação GET do recurso MyDictionary

3) Symbol: O recurso Symbol é utilizado para realizar operações sobre os símbolos de um dicionário. A única operação permitida é a de leitura, sendo a criação e edição, bem como a remoção de símbolos não abordadas no contexto deste trabalho. A Tabela V mostra a URI relativa desse recurso e as operações que ele suporta.

URI	POST	GET	PUT	DEL
/symbol/{MyDicionaryId}		X		
/symbol/{MyDicionaryId}/{name}		X		

Tabela V
OPERAÇÕES SUPORTADAS PELO RECURSO SYMBOL

No Exemplo 10 é possível visualizar o modelo de resposta da operação GET da URI "/symbol".

```
[
  {
    "name": "string",
    "comment": "string"
  }
]
```

Exemplo 10. Resposta da operação GET do recurso Symbol

4) Paradigm: O recurso Paradigm é utilizado para realizar operações sobre os paradigmas de um dicionário. Assim como o recurso Symbol, a única operação suportada no contexto deste trabalho é a de leitura. A Tabela VI mostra a URI relativa desse recurso e as operações que ele suporta.

No Exemplo 11 é possível visualizar o modelo de resposta da operação GET da URI "/paradigm".

URI	POST	GET	PUT	DEL
/paradigm/{MyDicionaryId}		X		
/paradigm/{MyDicionaryId}/{name}		X		
/paradigm/{MyDicionaryId}/suggestion		X		

Tabela VI
OPERAÇÕES SUPORTADAS PELO RECURSO PARADIGM

```
[
  {
    "name": "string",
    "entries": [
      {
        "left": "string",
        "right": "string",
      }
    ]
  }
]
```

Exemplo 11. Resposta da operação GET do recurso Paradigm

5) Lemma: O recurso Lemma é utilizado para para realizar operações sobre os lemas do dicionário. A Tabela VII mostra a URI relativa desse recurso e as operações que ele suporta.

URI	POST	GET	PUT	DEL
/lemma/{MyDicionaryId}	X	X		
/lemma/{MyDicionaryId}/{lineNo}		X	X	X

Tabela VII
OPERAÇÕES SUPORTADAS PELO RECURSO LEMMA

No Exemplo 12 é possível visualizar o modelo de resposta da operação GET da URI "/lemma".

```
[
  {
    "name": "string",
    "root": "string",
    "paradigm": "string",
    "lineNo": number,
    "inflections": [
      "value": "string",
      "symbols": [
        "string"
      ]
    ]
  }
]
```

Exemplo 12. Resposta da operação GET do recurso Lemma

B. Especificação da GUI

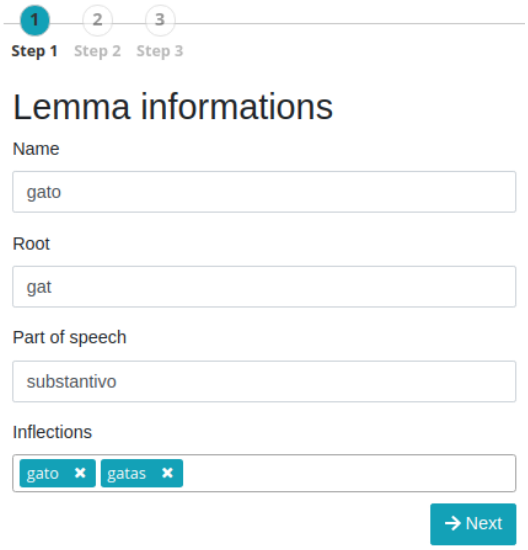
A seguir, é apresentado a proposta de uma interface, focada em usuários leigos em computação, para inserção de novos lemas em um dicionário morfológico.

A operação de inserção de um lema foi dividida em três passos. A Figura 4 apresenta o primeiro desses passos, que consiste na definição do nome do lema e de sua parte

invariante (*root*), seguido de duas informações opcionais: a classe gramatical do lema e alguns exemplos de flexão, que serão utilizados para encontrar sugestões de possíveis paradigmas para esse lema.

Foi utilizado como exemplo a inserção do lema "gato"em um dicionário morfológico de português. As informações inseridas foram: sua parte invariante, o prefixo “gat”; sua classe gramatical, substantivo; e duas de suas possíveis flexões: “gato” e “gatas”.

Figura 4. Primeiro passo na inserção de um lema



No segundo passo, apresentado na Figura 5, ocorre a apresentação dos possíveis paradigmas ao usuário. Quanto mais precisas forem as informações apresentadas na etapa anterior, menor será a quantidade de paradigmas sugeridos. O usuário deverá, então, verificar qual das sugestões melhor se encaixa ao lema. Na tabela inferior da imagem, é apresentado todas as flexões e seus respectivos símbolos do lema em questão utilizando o paradigma sugerido selecionado na tabela superior.

No exemplo utilizado, as informações inseridas no passo anterior resultaram na sugestão de apenas um paradigma, denominado “adept/o__n”. Ao selecioná-lo, são mostradas todas as flexões do lema “gato” utilizando-se esse paradigma. Cabe ao usuário verificar se cada flexão condiz com seus símbolos, e caso haja mais de uma sugestão, escolher a correta.

Na terceira e última etapa, apresentada na Figura 6, o usuário deve apenas confirmar as informações e, ao finalizar, o novo lema será inserido em seu dicionário.

C. Fluxo de trabalho

A arquitetura apresentada pode ser integrada à atual forma de desenvolvimento utilizando-se o fluxo de trabalho apresentado na Figura 7. Através do WDM, o usuário realiza

Figura 5. Segundo passo na inserção de um lema

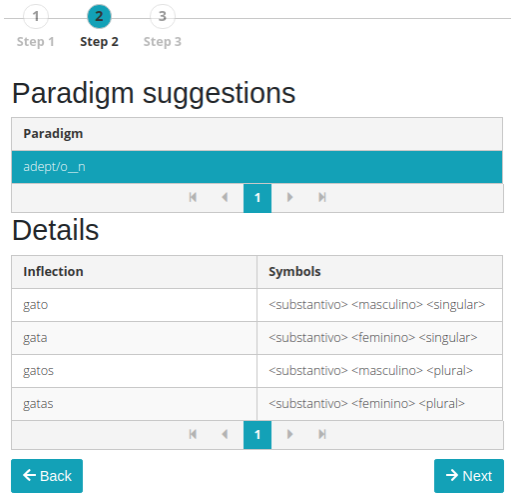
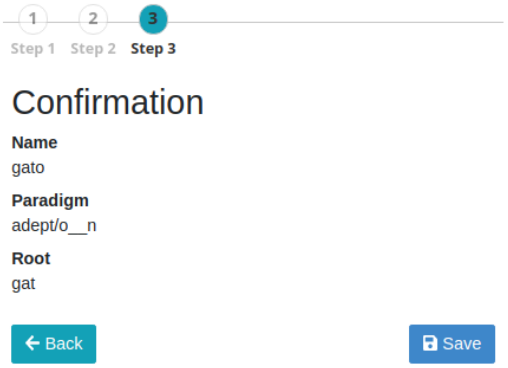


Figura 6. Terceiro passo na inserção de um lema



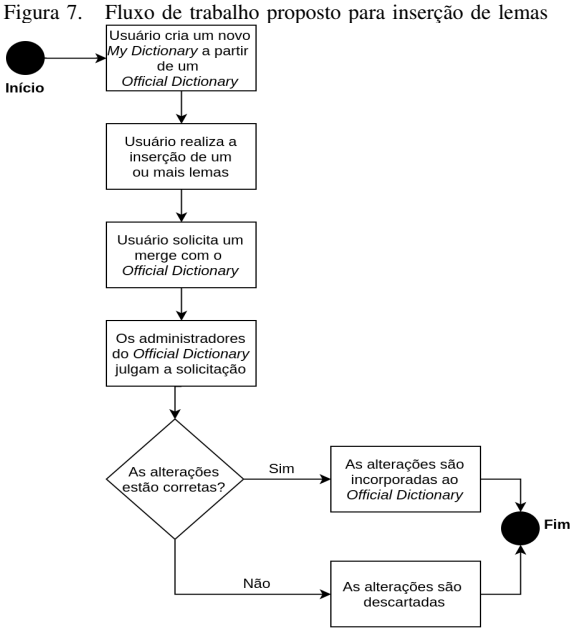
as três primeiras etapas apresentadas, que consistem na criação um novo dicionário para o usuário (uma *branch* no repositório Git); na inserção de lemas (*commits* individuais na *branch* do dicionário criado); e por fim no envio das contribuições para avaliação (*merge request* entre a *branch* criada e a *branch* principal). As etapas subsequentes já estão presentes no fluxo de trabalho atual, onde os mantenedores das bases de dados recebem as contribuições através de *merge requests* e verificam se as alterações devem ser aceitas ou não.

VI. Considerações Finais

Neste trabalho, foi realizado um estudo bibliográfico sobre a tradução automática. Nessa área, a tradução automática baseada em regras foi apontada como uma forte alternativa para realização de tradução automática entre línguas com baixa quantidade de *corpus* disponível.

O Apertium foi apresentado como uma das principais soluções de *software* livre para o desenvolvimento colaborativo de sistemas de tradução automática baseada em regras e foi

Figura 7. Fluxo de trabalho proposto para inserção de lemas



exposto que dentre suas principais limitações está o alto custo no desenvolvimento de seus recursos linguísticos.

Visando superar as dificuldades do desenvolvimento destes recursos linguísticos, foi proposto um ambiente *web* colaborativo que permite com que usuários não especialistas em computação possam contribuir com a expansão dos dicionários morfológicos utilizados no Apertium, de forma integrada à sua atual forma de desenvolvimento e sem precisarem adquirir conhecimentos específicos de computação.

O protótipo funcional desenvolvido contempla apenas as operações essenciais para adição de novos lemas aos dicionários morfológicos. Entretanto, a arquitetura pode ser facilmente expandida, de forma que permita realizar um conjunto maior de operações sobre os demais objetos desse dicionário e até mesmo para outros tipos de dicionário, como o bilíngue.

Dessa forma, o objetivo geral deste trabalho foi alcançado, uma vez que a arquitetura proposta mostrou-se como uma boa alternativa para abstrair os conhecimentos em computação necessários para uma pessoa se tornar apta a contribuir com a expansão dos dicionários morfológicos do Apertium. Ademais, o presente trabalho emerge como uma passo inicial para o desenvolvimento de um ambiente específico mais abrangente para desenvolvimento colaborativo e gerenciamento completo das bases de conhecimento do Apertium, além de permitir que outros desenvolvedores criem módulos editores para essas bases de conhecimento do Apertium utilizando os recursos disponibilizados pela API proposta.

REFERÊNCIAS

[1] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[2] C. Armentano Oller, A. M. Corbí Bellot, M. L. Forcada, M. Ginestí Rosell, M. A. Montava Belda, S. Ortiz Rojas, J. A. Pérez-Ortiz, G. Ramírez Sánchez, and F. Sánchez-Martínez, "Apertium, una plataforma de código abierto para el desarrollo de sistemas de traducción automática," 2007.

[3] A. Miranda, "Wiklats – um ambiente de interface e interaÇÃo para manipulaÇÃo e formalizaÇÃo de conhecimento," Curitiba, p. 80, 2009.

[4] G. RamirÉz-SÁnchez, F. Sánchez-Martínez, S. Ortiz-Rojas, J. A. Pérez-Ortiz, and M. L. Forcada, "Opentrad apertium open-source machine translation system: an opportunity for business and research," 2006.

[5] F. Tyers, F. Sánchez-Martínez, S. Ortiz-Rojas, and M. L. Forcadaac, "Free/open-source resources in the apertium platform for machine translation research and development," *The Prague Bulletin of Mathematical Linguistics*, vol. 93, 01 2010.

[6] F. M. Tyers, H. A. i Font, G. Fronteddu, and A. Martín-Mord, "Rule-based machine translation for the italian–sardinian language pair," 2017.

[7] R. Johnson, T. A. Pirinen, T. P. F. Tyers, T. Trosterud, and K. Unhammer, "North sámi to finnish rule-based machine translation system," 2017.

[8] F. Klubička, G. Ramírez-Sánchez, and N. Ljubešić, "Collaborative development of a rule-based machine translator between croatian and serbian," in *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, 2016, pp. 361–367.

[9] H. de Medeiros Caseli, "Tradução automática: estratégias e limitações," 12 2017.

[10] D. M. Eberhard, G. F. Simons, and C. D. Fennig. (2019) Ethnologue: Languages of the world. [Online]. Available: https://www.ethnologue.com

[11] M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers, "Apertium: a free/open-source platform for rule-based machine translation," *Machine Translation*, vol. 25, no. 2, pp. 127–144, Jun 2011. [Online]. Available: https://doi.org/10.1007/s10590-011-9090-0

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2015.

[13] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, and M. Norouzi, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[14] M. L. Forcada, B. I. Bonev, J. A. P. Ortiz, G. R. Sánchez, F. S. Martínez, C. Armentano-Oller, M. A. Montava, and F. M. Tyers, "Documentation of the open-source shallow-transfer machine translation platform apertium," 2010.

[15] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Berkely, CA, USA: Apress, 2014.